NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AMPHIBIOUS OPERATION SIMULATION

by

Chanok Hongnoi
December 1982

Thesis Advisor: Norman Lyons

Approved for Public Release; Distribution Unlimited

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): Amphibious Operation Simulation	5. TYPE OF REPORT & PERIOD COVERED: Master's Thesis December, 1982
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Chanok Hongnoi	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School Monterey, California 93940	12. REPORT DATE: December, 1982
	13. NUMBER OF PAGES: 96
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report): unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer Simulation, Ship-to-Shore Movement Control, Programming Amphibious Operation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Since the price of personal computers is coming down, it is possible to have computers in a small ship that has a limited budget. The commanding officer of a small ship needs a support system for making decisions in amphibious operations. A personal computer would be helpful in saving time manipulating the information used to make decisions in amphibious operations.

(Continued)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 68 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ABSTRACT (Continued) Block # 20

The area to be investigated is the approach to the systems analysis and design of the amphibious operation controller and simulation program in ship-to-shore phase. We use the computer to control the waves of small boats that carry the troops to shore.

Approved for public release, distribution unlimited.

Amphibious Operation Simulation

by

Chanok Hongnoi
Lieutenant, Royal Thai Navy
B.S., Thai Naval Academy, 1974

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
December 1982

ABSTRACT

Since the price of personal computers is coming down, it is possible to have computers in a small ship that has a limited budget. The commanding officer of a small ship needs a support systems for making decisions in amphibious operations. A personal computer would be helpful in saving time manipulating the information used to make decisions in amphibious operations.

The area to be investigated is the approach to the systems analysis and design of the amphibious operation controller and simulation program in ship-to-shore phase. We use the computer to control the waves of small boats that carry the troops to shore.

TABLE OF CONTENTS

I. INTRODUCTION 9

A. AMPHIBIOUS OPERATIONS 9

1. General 9

2. Definition and Characteristics 10

3. Purpose 11

4. Sequence 11

5. Planning 11

6. Embarkation 12

7. Rehearsal 12

8. Movement 13

9. Assault 13

10. Assault Operation 13

B. SHIP-TO-SHORE MOVEMENT 14

1. General 14

C. CONTROL OF SHIP-TO-SHORE MOVEMENT 16

1. General 16

2. Area Organization for Control Landing . . 17

II. SYSTEM SURVEY. 20

A. AMPHIBIOUS OPERATION 20

B. SYSTEM DESIGN 23

1. Subroutine Print Character 23

2. Subroutine Print Message 24
3. Subroutine Numeric Input 24
4. Subroutine Set Character 24
5. Subroutine Plot Grid 25
6. Subroutine Time Scale 25
7. Subroutine Wave Position 26
8. Subroutine Print Time 27
9. Subroutine Read Time 27
10. Subroutine Error Table 27
11. Subroutine Error Handler 28
12. Subroutine Compute Time Difference . . . 28
13. Subroutine Input 29
14. Subroutine Grid Report 29
15. Subroutine Echo Input 29
16. Subroutine Offset 30
17. Subroutine Master Control 30

III. USER'S MANUAL. 31
A. SPECIFICATION. 31
B. USE OF THE PROGRAM. 31
C. ERROR MESSAGES SUMMARY. 37
D. CONCLUSION. 38

APPENDIX A: ALGORITHM. 39

APPENDIX B: PROGRAM LISTING. 61

APPENDIX C: COMMAND SUMMARY. 84

LIST OF REFERENCES 95

INITIAL DISTRIBUTION LIST 96

LIST OF FIGURES

2.1. Compute time 21

3.1. Displayed Grid 32

3.2. Simulation and Control Display 35

3.3. End of Control 36

I. INTRODUCTION

A. AMPHIBIOUS OPERATIONS

In World War II, at Normandy on of the French coast, 132,715 troops were landed in sixteen hours on 6 June 1944 against some of the most sophisticated shore defences then known. At Okinawa, an island about 600 miles south of Japan, over 183,000 men in 1,300 vessels made the last a succession of major landings by United State amphibious force against determined Japanese defenders in April 1945.

1. General

Amphibious warfare integrates virtually all types of ships, aircraft, weapons and landing forces in a concerted military effort against a hostile force.The salient requirement of the amphibious operation is the necessity of building up combat power ashore from an initial zero capability to fully coordinated striking power as the attack drives toward the final objectives. The amphibious assault must be conducted in the face of certain additional difficulties. Natural forces such as unfavorable weather, seas, surf, and

features of hydrography represent hazards not normally encountered in land warfare. Technical problems of logistics include loading thousands of troops and large quantities of material into ships at widely separated embarkation points, moving them to the objective, and then landing them in exactly the proper sequence. Usually on open beaches or landing zones and under fire initially. All this requires extraordinary attention in the form of detailed planning. During the movement from ship-to-shore, troops are especially vulnerable. Possible employment of mass destruction weapons by the enemy is a threat to the amphibious task force, as to any other offensive concentration and requires the exercise of effective countermeasures, both active and passive, during the stages of the operation.

2. Definition and Characteristics

An amphibious operation is an attack launched from the sea by naval and landing forces embarked in ship or craft involving a landing on a hostile shore. It normally requires extensive air participation and is characterized by closely integrated efforts of forces, trained, organized and equiped for different combatant function.

3. Purpose

Amphibious operations are conducted primarily to establish a landing force on a hostile shore in order to;

(1) Prosecute further combat operations

(2) Obtain a site for an advanced naval or airbase; and

(3) Deny the use of an area of facilities to the enemy.

4. Sequence

The amphibious assault follows a well defined pattern. It includes a sequence of events or activities, which occur, although to a lesser degree, in other types of amphibious operations. The general sequnce consists of planning embarkation, rehearsal, movement to the objective, and finally assault and capture of the objective. Planning, for example, occurs throughout the entire operation but is dominant only in the period prior to embarkation.

5. Planning

The planning phase denotes the period extending from the insurance of the initiating directive to embarkation.

During this phase, the necessary preparatory measures, including coordinate planning are effected. Although planning does not cease with the termination of this phase, it is useful to distinguish between the planning phase and the subsequent operational phase, since a marked change occurs in the relationship between the commanders of the various service components at the time the planning phase is terminated and the operational phases begin. At the commencement of the operational phases, the commander of the amphibious task force assumes full responsibility for the entire force and for the operation.

6. Embarkation

The embarkation phase is the period during which the forces, with the equipment and supplies, embark in assigned shipping.

7. Rehearsal

The rehearsal phase is the period during which the prospective operation is rehearsed for the purpose of:

(1). Testing the adequacy of plans, timing of detailed operations, and the combat-readiness of participating forces;

(2). ensuring that all echelons are familiar with plans; and

(3). testing communications.

8. Movement

In this phase, the components of the amphibious task force move from the points of embarkation to the objective area. This move may be via rehearsal, staging, and/or rendezvous areas. The movement phase is completed when the components of the amphibious task force arrive in their assigned position in the objective area.

9. Assault

The assault comprises the period between the arrival of the major assault forces of the amphibious task force in the objective area and the accomplishment of the amphibious task force mission. Development of the area for its ultimate use may be initiated during this period.

10. Assault Operation

Assault operations by the landing force begin with the ship-to-shore movement and the landing of the first schedule wave, and terminate with the capture of the final

ground objectives of the landing force. Other forces continue to provide logistic and fire support during the assault operation of the landing force and continue to provide over-all protection of the amphibious task force.

B. SHIP-TO-SHORE MOVEMENT

1. General

a. The ship-to-shore movement is that part of the assault phase which pertains to the timely deployment of troops and their equipment from assault shipping to designated positions ashore in the landing area. More precisely, this movement is designed to ensure the landing of troops, equipment, and supplies at the prescribed times and places and in the information required by the landing forces' plan of maneuver for operations ashore. The movement may be executed by waterbrone means (landing ship, landing craft, and amphibious vihicles), by helicopters, or by a combination of the two.

b. Although the ship-to-shore movement is only a part of the assault phase, it is the most critical part. The achievement of the requisite coordination and control of the many diversified naval and troop elements participating in the ship-to-shore movement imposes tasks which are unparalleled in scope by anyother military operation in modern warfare. Ship-to-shore movement planning reflects

to a preeminent degree the requirement for concurrent and parallel planning at all naval and troop echelons. The landing plan, consisting of a variety of documents, must leave no doubt as to what is intended. Whereas, generally in military operations, subbordinate commmanders are only told what to do, there is, in the ship-to shore movement and equally compelling requirement to spell out many of the details concernning how the operation will be accomplished.

c. The ship-to-shore movement may encompass any or all of the following operations:

(1) Assembly of landing ship, landing craft, amphibious vehicles and helicopters in required formation for debarkation and landing.

(2) Debarkation of troops, equipment, and supplies from ship into appropriate ship-to-shore movement means;

(3) Transfer operations

and,

(4) Controlled landing of the assault echelons of the landing force to includes equipment and supplies.

d. The ship-to-shore movement commences on order of the amphibious task force commander after consultation with the landing force commander. The movement is brought to a close when all of the troops, equipment, and supplies loaded in assault shipping have landed.

e.The ship-to-shore movement is divided into two periods.

(1) The initial unloading period which is primarily tactical in character with emphasis on responsiveness to landing force requirements ashore. It is during this period of the ship-to-shore movement that the landing force is graining its foothold ashore, and the unloading and landing of essential men, equipment, and supplies is very selective.

(2) The general unloading period, which is primarily logistical in character, emphasizes to the delivery of quantity in the shortest time possible. Although the aim shifts to volume and speed rather than selectivity, the unloading and landing operation should never exceed the capability of logistic operations ashore to handle that which is delivered.

C. CONTROL OF SHIP-TO-SHORE MOVEMENT

1. General

Control of the ship-to-shore movement directly or indirectly, involves the organization of the sea area, the establishment of a number of control agencies, and the employment of various control techniques and devices. The number and arrangement of beaches and the corresponding types of movement also play a part in the control system to be established for a particular operation.

2. Area Organization for Control Landing

For a discussion of the organization of the sea area in its entirety. Organization of the sea operation area close to the shore for execution of the ship-to-shore movement involves certain coordination and control devices as discussed below.

(1) The line of departure is an off-shore coordinating line, approximately parallel to the landing beach, from which the successive waves of boats are dispatched for their final movement to the beach. If beaches are separated, each beach has its own line of departure, which is marked by a ship or ships of the control organization. The location of the line of departure is governed by topographic, hydrographic, and tactical considerations.

(2) Boat lanes extend seaward from landing beaches of the line of departure. The width of the boat lanes is determined by the length of the coresponding beaches.

(3) Approach lanes are extensions of boat lanes from the line of departure toward the transport ares. They may be terminated by marker ships, boats, or bouys. Adjacent approach lanes may be parallel or may diverge to seaward to provide for early dispersion of the waves of boats.

(4) A floating dump area is an off-shore area in which are stationed a designated number of landing craft or

amphibious vehicles, loaded with supplies to meet early demands of the trocps ashore. A floating dump area should be located in the vicinity of the line of departure, with due regard for adequate dispersion and ease in control.

(5) Special unloading berths into which transports may move for unloading are established in the vicinity of the approach lanes. This result in reduction of the running time of landing craft and amphibious vehicles and assists in the dispersion of transports.

(6) A casualty evacation control berth is established for a ship which may be specially equiped for handling casualties. Usually this is a landing ship in which a casualty evacuation control officer is embarked. Normally one berth is allctted to each beach. A berth is established to serve one or more landing beaches, depending upon the proximity of landing beaches, and is located as close to the beach as condition permit.

(7) The transfer area is a designated ares to seaward of the surfline, off a landing beach, where personnel and material are trasfered from landing craft to amphibious vehicles. It is established when troop plans, terrain, or hydrographic condition dictate.

(8) A transfer berth is located off a landing beach in the proximity of the transfer lane. A crane-equipped ship or a barge is stationed here to transfer troops, suplies, and equipment frcm landing craft to amphibious vehicles.

(9) The amphibious vehicle launching area is a designated area located in the near vicinity and to seaward of the line of departure. The ships carrying amphibious vehicles move into this area to unloaded them. The area is so located in relation to the line of departure as to ensure a minimum amount of maneuver and sea area transits by the amphibious vehicles prior to crossing the line of departure.

(10) The causeway lanching area is an area located near the line of departure but normally clear of the approach lanes, where ships can launch pontoon causeways. This area is so located that the causeway can be launched in a minimum amount of time and with least interference from other unit operating in the immediate area. Causeways are located adjacent to but not in boat lanes, usually toward the flank of the beach.

II. SYSTEM SURVEY.

A. AMPHIBIOUS OPERATION

The purpose of this study is to simulate the control of amphibious operation in the ship-to-shore-movement phase. The problem is how to get the landing force to the shore on time. The time here is defined in term of D-Day, H-Hour, M-Minute, which means that at the date of D-Day the time of H-Hour and M-Minute, the landing force will do the amphibious operation. They reach the shore by that time, not before or after. In order that landing groups reach the shore on time, the course and speed of the landing groups have been previously defined. Usually, the course is perpendicular to the shore and the speed is usually 70%-80% of the maximum speed of the landing craft. The landing groups are parts of the whole landing force. The landing groups are also divided into waves when landing which are controlled by wave guide commander. Those wave guide commanders are controlled by the control ship. Since the control ship may be destroyed at any time. It is therefore, reasonable to have 2 control ships called the Primary Control Ship (PCS) and the Secondary Control Ship (SCS) both ships can keep track of the movement of Wave Guide Commander (WGC), but the Primary Control Ship (PCS) controls the Wave Guide Commander (WGC). If the

Primary Control Ship (PCS) is destroyed, the Secondary Control Ship will take over the control. The Primary Control Ship (PCS) and Seconary Control Ship are usually anchored 2,000 yards away from the shore at the other side of the boat lanes. The boat lanes are the path of the waves. The width of the lanes is about 400 yards the course is the wave course and the length is the distance between the shore and control ships, usually 2,000 yards. The line drawn between the Primary Control Ship (PCS) and Seconary Control Ship would be across the boat lane and parallel to the shore. This line is called Line of Departure (LoD). By using the wave's speed, we can compute the time the wave was at the Line of Departure (LoD).

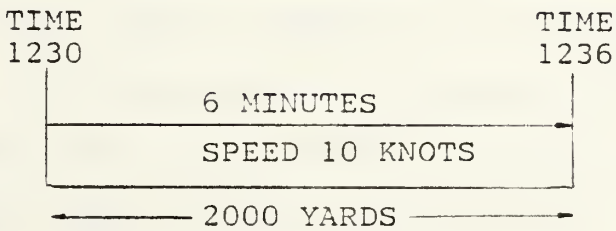

Figure 2.1 Compute time

For example, if the boat's lane's length = 2,000 yards wave speed = 10 knots, H-Hour = 12, M-Minute =30. It would take 2000*60/(10*2000) = 6 minutes (1 mile = 2,000 yards) for the wave to reach the shore. Therefore, that the time to pass Line of Departure (LoD) would be 12:24.

Five minutes before the time to pass Line of Departure (LoD), the Primary Control Ship (PCS) raises a Zero flag in order for the waves to proceed to the Line of Departure (LoD). At the time to pass Line of Departure (LoD) or 12:24, as the example above, the Primary Control Ship (PCS) haults down the flag while the waves forward to the shore it will report the grid position every minute. The grid position is based on whether the Wave Guide Commander (WGC) is in the middle left or right side of the boat lane. We divide the boat lane into 5 regions, Hence the boat lane that is 400 yards wide, 0- 50 yards from left is labeled "LL", 50-100 yards from left is labeled "L", 100- 200 from left is labeled "C", 0-50 from right is labeled "RR", 50-100 from right is labeled "R". The report includes whether the Wave Guide Commander (WGC) is early , late, or on time. If it is early, the wave should decrease the speed. When the waves reach the shore the Wave Guide Commander (WGC) will report to the Primary Control Ship (PCS) "touch down" at which time the Primary Control Ship (PCS) will no longer control the wave.

B. SYSTEM DESIGN

According to the characteristic of computer used in this study, we have to built our own characters when we use the high resolution mode. The characters have been previously designed. Each character occupies 5 bytes of memory. From the program listing, each of the data statements contains the data for one character. The line number of the statement minus 2,200 is the ASCII code value for the character. Each character is defined by 5*7 array of points. Each point in the array is translated into the paper bit pattern, and the resulting numbers are placed in the data statement. The data are then stored in memory from location 31647 by subroutine SET CHARACTER.

1. Subroutine Print Character

This routine is used to print the character at the position ZX, ZY where ZX is 256 horizontal does on the screen when this routine is called. ZZ$ is the passing parameter, ZZ$ is the character we want to be printed. The function ASC(ZZ$) will return ASCII value to ZC. The statement ZZ = ZC*5+31647 will assign the table pointer value to ZL. ZQ is the variable pointer of an array A. This allows the character data to be stored in variable A. A, then, will be PUT to a specific location on the screen specified by ZX,ZY.

2. Subroutine Print Message

This routine is used to print a string of characters on screen. the first character will be at location specified by ZX and ZY. This will be accomplished by moving the consecutive character into ZZ$ and then call PRINT CHARACTER routine.

3. Subroutine Numeric Input

This routine work like INPUT statement in BASIC, but the prompt message can be located by ZX and ZY. This routine while waiting for the input, will keep on reading the time from the real time clock. It also checks input data, if the input data is not numeric (ASCII<48 or >57 in decimal or $30-$39 in hexadecimal).

4. Subroutine Set Character

This routine reads the character data from the program and place it in the physical memory location from the address 31647 tc the adiress 32125 which is the highest location for 32K computer.

5. Subroutine Plot Grid

This routine is to plot the boat lanes grid on the screen by using 10 yards:1 dot on the screen the first do loop is to draw 6 horizontal line which are LL, L, C, R, RR regions. The next DO loop draws vertical distance scale from 0 to 24000 yards for every 200 yards. Label numbers are in hundred of yards. The sign " " is the location of Primary Control Ship (PCS) specified by PX and PY which are the input offset values from the INPUT routine. The rest of this routine prints all the labels such as "SHORE" draws arrow, print course etc..

6. Subroutine Time Scale

In order to plot the time scale on boat lanes in every minute, we have to know the speed, then convert it to distance every minute and then draw a correspond line on the screen

SC = number of dots on the screen (10 yards per dot)

SC = SP*2000/(60*60) yards/sec

=SP*2000(60*60) dots/sec

=SP*2000*20/(60*60*10) dots for every 20 seconds

A DO loop is used to label the time scale every minute.

7. Subroutine Wave Position

This routine is the routine which interacts with the user. We have to make sure that error will not occur. This routine will ask the user to reinput the data if the data fails certain test. This routine will get the input bearing and range first. It will get bearing by printing the "?" after the word "bearing". The input bearing will be assigned to variable BR first, while loop check if the input bearing is between 0 and 360. If it is not, then this routine will keep on asking for the input. After the computer gets the correct bearing, the "?" will be printed after the word "RANGE" the user is supposed to input range of the wave guide from the Primary Control Ship (PCS). Again this routine will check the user input numeric data. Only after the computer gets the correct input. The true bearing will be computed by usingthe equation

```
WH=(BR-CO)/180*3.141592654
WX=INT(PX-RG/10*COS(WH))
WY=int(PY-rg/1)*SIN(WH))
```

To change bearing and range respect to offset to dot position on the screen. This routine also provides the mechanism that allow us not to plot out off screen edge. For the first input a dot (position of wave) will be ploted on the screen using the command PRESET(WX,WY) (since the screen

is usually SET by the command PCLS1). After the first input, the output position will be lined to the previous one. The position is marked by the "<" sign in the GET and PUT command. The GET command will get every thing at position. The "<" will be printed (the area cover 5*8 dots) and will be PUT back later after the computer compute a new position and then GET the data from the new position before printing "<" and so on.

8. Subroutine Print Time

This routine is to print Hour minute and second of the present time. This routine will be called once after each input. Therefore, the time output is latched.

9. Subroutine Read Time

While waiting for bearing and range to be inputs, this routine will be called repeatedly, together with polling keyboard in order to update the time variable TH,TM,TS (note that the real-time-clock is still running all the time by using 1/60 second hardware interupt).

10. Subroutine Error Table

This routine only handles error messages.

11. Subroutine Error Handler

This routine outputs the error meaasges into error message area on the screen when this routine is called only the number of error message is passed.

12. Subroutine Compute Time Difference

TU is the time taken to reach the shore at speed SP.

TU=WX*10*60/(2000*SP)

=WX*.3/SP

TP is the planned time in minute, if TP= 4 means by the schdule, we have 4 minutes to reach the shore and if TU=2, means it would take only 2 minute to reach the shore, if we retain the same speed, another word we can say is that we are 4-2 =2 minutes earlier than we have planned. This is how we can determine if we are early, late, or on time. Hence it takes about 1 or 2 seconds to print a message. If the message is the same as the last time we have printed on the screen, we don't have to put it again on the screen the messages will stay latch until we print blanks over them. To handle this we use TT$ to be the value of the last message.

13. Subroutine Input

This routine use normal screen (alphanumeric mode) to input, in order to minimize the number of statements. This routine input H-Hour(HH), M-Minute,time in hour(TH) time in minute (TM), course of wave (CO), speed of wave (SP) and also set real-time-clock, each of the input is also checked for validation.

14. Subroutine Grid Report

This routine use the value of WY (wave position of Y-axis) and determine whether the WGC is in which region by returning the value of grid "LL","L","C","R","RR"to gr$ and we use GC$ to store the value of the recommened vector, GG$ store the value of the last vector output. If the current vector output is the same as the last one, the output will not have to be printed again.

15. Subroutine Echo Input

This routine just echoes the input by using the high resolution mode. The inputs echoed are wave speed H-Hour, M-Minute, also prompt bearing, range, and display all labels.

16. Subroutine Offset

This routine get the position of Primary Control Ship(PCS) by using the reference position (the point on the left edge of boat lane intersected by the Line of Departure (LoD)) or column 200 row 172 on the physical location on the screen. PX, PY will be the physical location of Primary Control Ship(PCS) on the screen.

17. Subroutine Master Control

This routine just calls the lower level modules and prompts the user if he would like to continue.

III. USER'S MANUAL.

A. SPECIFICATION.

This program is designed to use with TRS 80 Color Computer 32 K bytes with Extended BASIC system. The program size is 11 K bytes and need working area for at least 1 K bytes. Since the monitor system occupy 8 K bytes and we use 4 graphic pages or 6 K bytes. Therefore this system consumes 26 K (6+8+1+11) of memory space. The machine that has memory less than 32 K bytes is not recommened.

B. USE OF THE PROGRAM.

This program is on the cassette tape which should be loaded with the CLOAD"AMPHI" command. When the program is already loaded, simply type RUN to run the program, you have to wait for only a few seconds. You will be prompted for a series of parameters as shown below.

H-HOUR	the hour, time when the landing force were at the shore.
SET HOUR	the present hour.
M-MINUTE	the minute, time when the landing force were at the shore.
SET HOUR	the present hour.
SET MINUTE	the present minute.

BOATLANE'S COURSE the course which the wave planed to use usually perpendicular to the shore.

WAVE'S SPEED the speed which the wave planed to move.

OFFSET BEARING the bearing from the planed PCS.location to the real location.

OFFSET RANGE the range from actual PCS.location to therotical location.

After you have answer all the questions, it will take a while to plot the grid and all labels as shown in figure 3.1

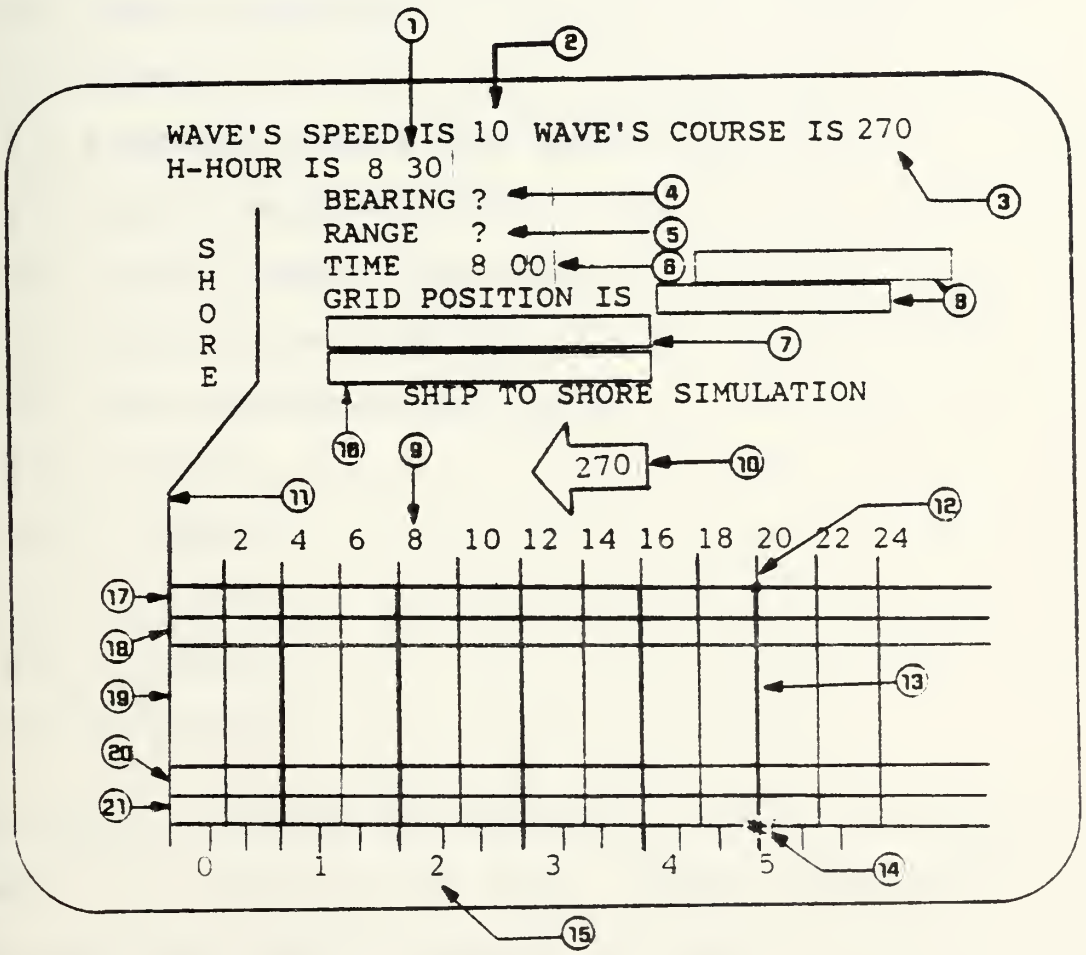

Figure 3.1 Displayed Grid

1. Echo H-Hour input.
2. Echo wave's speed input.
3. Echo wave's course input.
4. Bearing input area.
5. Range input area.
6. Display time.
7. Control command area.
8. Grid position display area.
9. Distance from shore in hundred of yards.
10. Wave's direction.
11. Shore.
12. Secondary Control Ship (SCS).
13. Line of Departure (LoD).
14. Primary Control Ship (PCS).
15. Time's scale in minute.
16. Error message display area.
17. RR region.
18. R region.
19. C region.
20. L region.
21. LL region.

After grid is plotted the vertical line will be labeled by numeric. On the top of the grid, numbers indicate the distance from the shore in hundreds of yards. The Line Of Departure will be at 2,000 yards from the shore or labeled 20.

The rendezvous area will be anywhere beyond the 2,000-yard line. The label at the bottom is the time scale in minutes which corresponds to your input wave's speed. If the wave speed were 10 knots, the time required from the Line Of Departure (LOD.) to the shore would be 6 minutes. When you see the "?" after the word "BEARING", then you can input bearing in degrees. The maximum input is 360, if you input the number larger than that an error message will be displayed. After you have input bearing you will see the "?" again, after the word "RANGE"., You may input the range of the wave guide from Primary Control Ship (PCS). Usually, we can get the bearing and range from RADAR. Within a second the screen will display time. The time will stay latched until you input another bearing and range. You will see the position of the wave guide indicated by "<" sign anywhere on the boat lanes depends on your inputs, the screen also displays the grid position indicated by the letters C,L,LL,R,RR which mean that on the middle, on the left, on the far left, on the right, on the far right respectively. For example, if the grid position is L, it means you are on the left side of boat lanes and a change of course to the right.

In this case, "VECTOR RIGHT 5" will be displayed as shown in figure 3.2. The computer tells you the vector every time you are not at "C" position. If you stay at the far right position in "LL" the vector to be recommened would be "VECTOR LEFT 10). The computer also tells you whether you

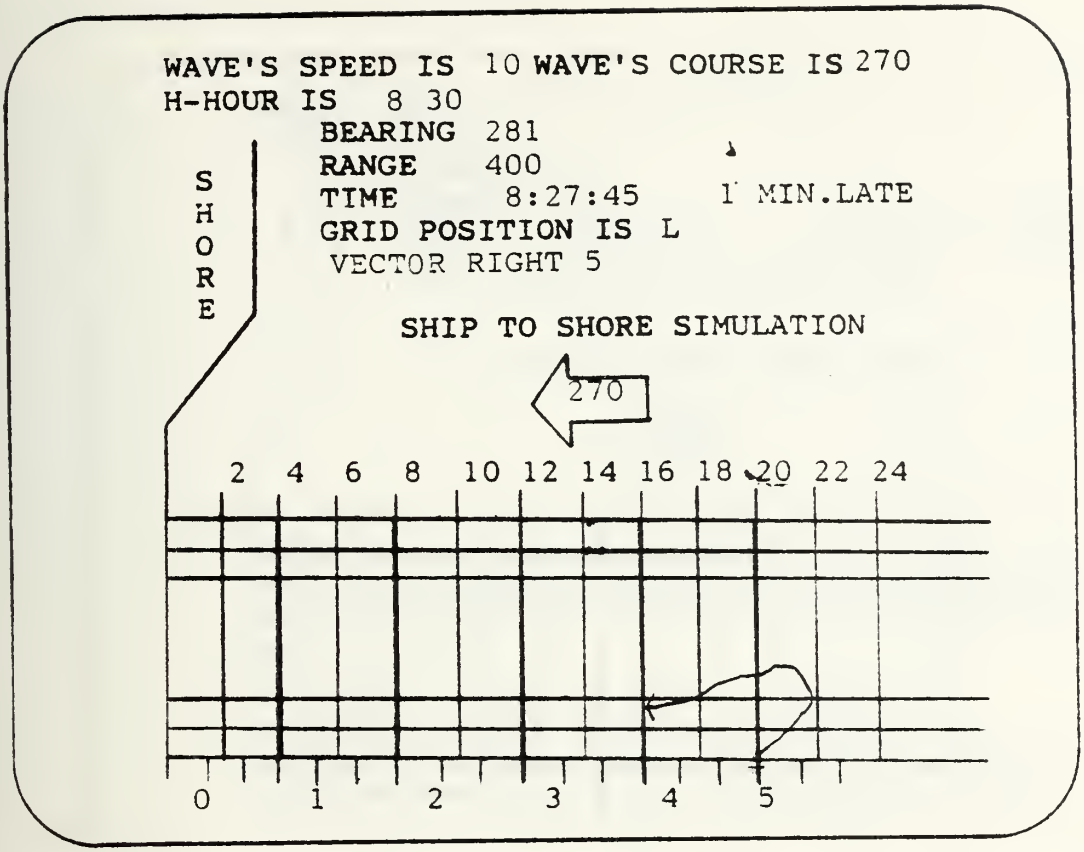

Figure 3.2 Simulation and Control Display

are on time or not. If you are late or early the computer will tell you how late or early you are. This message will be displayed after the grid position. For example, if you have 3 minutes to get to the shore and you are at 1 minute line, the display would be 2 minutes early, you are supposed to reduce the speed in order to get to the shore exactly on time. After your second bearing and range inputs, the output location of wave guide will be linked to the previous one. Therefore you can track the movement of the wave, enabling you to provide better decision making.

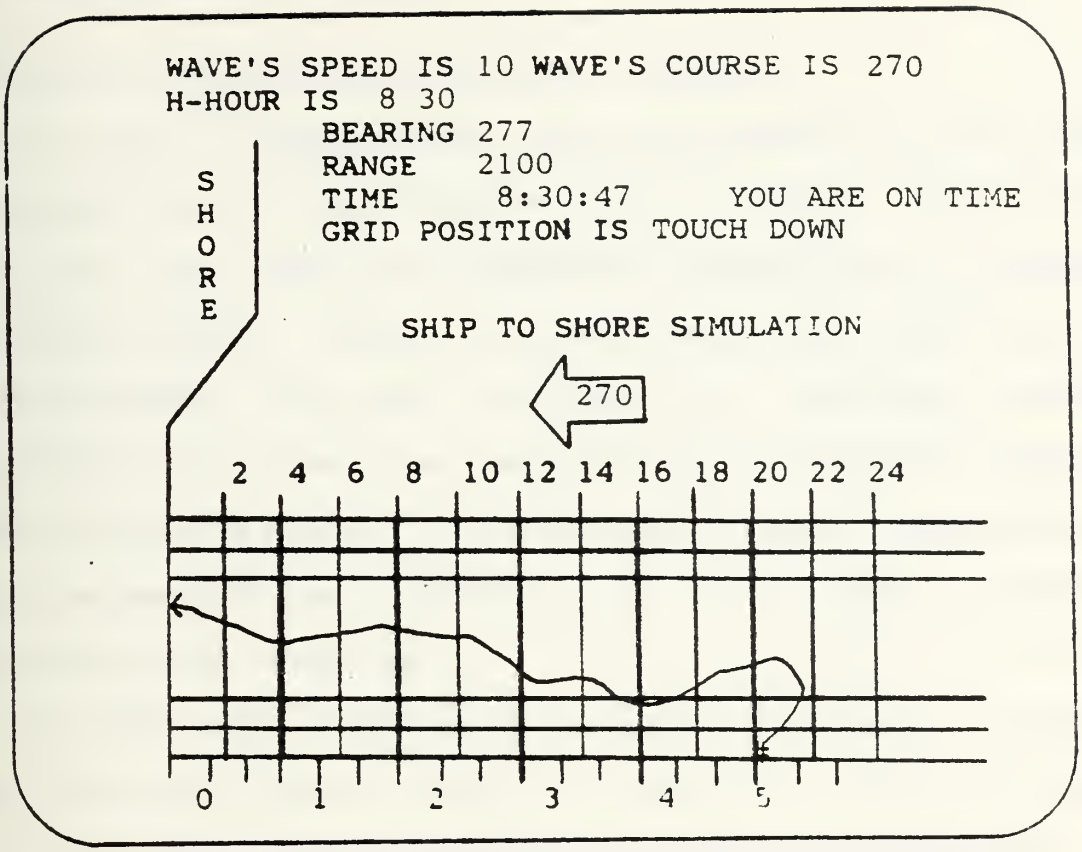

Figure 3.3 End of Control

Finally when you reach the shore the computer will display "TOUCH DOWN", the control is complete see figure 3.3, and you are now ready to control another wave.

C. ERROR MESSAGES SUMMARY.

There are 5 errors produced by the program.

1. OUT OF SCREEN error means the misposition of wave guide which the computer plots. Out of screen eror occurs when it is not within the computer's capability to provide output on screen. This error may result from the out-of-screen boundary data (the horizontal dot position greater than 255 or less than 0 or the virtical dot position greater than 191 or less than 0) or from wrong inputs, such as the range between the wave guide and PCS is too long or bearing is in the wrong direction.

2. TOO SMALL VALUE results from inputs which are too small, eg. when the wave speed is lower than 2 knots or when the input which should be positive turned to be negative.

3. GREATER THAN 360 happens only when dealing with bearing input. Since bearing can be at its most at 360 degrees, if bearing input more than 360 degrees, this error will result.

4. TOO LARGE VALUE results from inputs which are too large, eg. when we input time more than 23 hours, or 59 minutes; or when we input range more than 9,999yards.

5. INVALID INPUT results from wrong types of data, eg. alphabetical data is input in place of numerical data.

D. CONCLUSION.

The ship-to-shore movement is that portion of the assault phase which includes the employment of landing forces from the assault shipping to designates landing areas. It has as its purpose the landing of assault units at the proper times and places and in the formations required by the landing force plan of maneuver ashore. It commences on order of the amphibious task force commander and is brought to a close when unloading of assault shipping is completed.

In order to increase the ship-to-shore movement control efficiency, including accuracy of times and places, computer usage in controlling proves to be very benefical. Normally ship-to-shore movement control is based on manual plotting. This process takes approximately 30 seconds to one minute, from the time of receiving input, to the time of plotting the wave. The use of computer in control provides needed output within less than 5 seconds after the time of data input and control more boat lanes accurately in a few seconds. Additionally, the computer control system may be used in other phases of amphibious operation; eg. firing support system, rehearsal phase, etc.

In the near future, hundreds of thousands of lives may be saved if computer control system replaces the manual amphibious operation.

APPENDIX A

ALGORITHM.

(This program is to simulate the amphibious operation in ship to shore movement phase)

(varible declaration)

a$ = string buffer for display

c$ = numeric input buffer

gc$ = command used for control

gr$ = grid report buffer

q$ = quit command

sH$ = output the word "shore"

t$ = time report storage

tt$ = previous time report output

zz$ = character of a$

bp = offset bearing

br = bearing wave input

cl = lenght of c$

co = lain's course

em = error message location

fg = flag of position plot

g = varible grid plot

hh = h-hour

i = loop counter

j = loop counter

k = loop counter

l = length of a$

mm = m-minute

ph = direction in degree

px = horizontol axis of pcs.

py = vertical axis of pcs.

qq = quit flag

rg = range from wave guide

rn = offset range

sc = time scale

sp = speed of wave guide

tc = difference of wave time and predicted time

tp = predicted time

th = time in hour

tm = time in minute

ts = time in second

tu = actual time of wave

vx = previous horizontal axis of wave

vy = previous vertical axis of wave

wh = relative bearing of wave in degree

wx = horizontal axis of wave location

wy = vertical axis of wave location

xv = numeric input from key board

z4 = character width

z7 = character high

z8 = character gap

z9 = line gap

zc = character data

zl = data pointer to read character

zq = varible pointer of array "a"

zx = horizontal location of alphanumeric output

zy = vertical location of alphanumeric output

```
master control program
   call set characters
   call error table
   q$ = ""
   do while q$ <> "q"
      call input data
      call offset
      call echo input
      call set time scale
      call plot grid
      do while qq <> 1
         call wave position
         call print time
         call compute time difference
         call grid report
      end do
      input q$
   end do
end master control
```

```
subroutine offset
   input "the offset of primary control ship from reference
       point in bearing" ;bp
   do while bp > 360
       print er$(3)
       input "the offset of primary control ship from
              reference point in bearing" ;bp
   end do
   do while bp<0
       print er$(2)
       input "the offset of primary control ship from
              reference point in bearing" ;bp
   end do
   input "range from reference point";rn
   ph = int(bp-co)/180*3.141592654
   px = int(200-rn/10*cos(ph))
   py = int(172-rn/10*sin(ph))
   do while px > 255 or px < 0 or py > 191 or py < 72
      input "range from reference point";rn
      print er$(1)
      ph = int(bp-co)/180*3.141592654
      px = int(200-rn/10*cos(ph))
      py = int(172-rn/10*sin(ph))
   end do
return
```

```
 subroutine echo input
      (set screen to graphic mode)
      zx=0
      zy=0
      a$="wave's speed is" + str$(sp) + " wave's course is" +
str$(co)
      call print_message(zx, zy,a$)
      zx = 0
      zy = 9
      a$ = "h-hour is" + str$(hh) + ":" + str$(mm)
      call print_message(zx, zy,a$)
      zx=36
      zy=18
      a$="bearing"
      call print_message(zx, zy,a$)
      zx=36
      zy=27
      a$="range"
      call print_message(zx, zy,a$)
      zx=36
      zy=36
      a$="time"
      call print_message(zx, zy,a$)
      zx = 36
      zy = 45
      a$ = "grid position is"
      call print_message(zx, zy,a$)
```

```
    zx = 114
    zy = 18
    a$ = "degrees"
    call print_message(zx, zy,a$)
    zx = 114
    zy = 27
    a$ = "yards"
    call print_message(zx, zy,a$)
    return


subroutine grid report
  if wy < 142 and wy >= 137 then
        gr$ = "r "
        gc$ = "vector left 5 "
  end if
  if wy < 162 and wy >= 142 then
        gr$ = "c "
        gc$ = "                "
  end if
  if wy < 167 and wy >= 162 then
        gr$ = "l "
        gc$ = "vector right 5 "
  end if
  if wy >= 167 then
        gr$ = "ll"
```

```
      gc$ = "vector right 10 "
 end if
 if wy < 137 then
      gr$ = "rr"
      gc$ = "vector left 10 "
 end if
 if wx <=0 then
      gr$ = "touch down"
      zx = 32
      zy = 72
      a$ = "press'q' to quit else continue !"
      call print_message(zx,zy,a$)
      qq = 1
 end if
 zy = 45
 zx = 138
 a$ = gr$
 call print_message(zx,zy,a$)
   if gg$ <> gc$ then
      zy = 54
      zx = 36
      a$ = gc$
      call print_message(zx,zy,a$)
      gg$ = gc$
   end if
return
```

```
subroutine input data
   (clear screen)
   input"h-hour";hh
   do while hh > 23
      print er$(4)
      input"h-hcur";hh
   end do
   do while hh < 0
      print er$(2)
      input"h-hour";hh
   end do
   input "m-minute";mm
   do while mm > 59
       print er$(4)
       input "m-minute";mm
    end do
    do while mm < 0
       print er$(2)
       input "m-minute";mm
    end do
    input "set hour";th
    do while th > 23
       print er$(4)
       input "set hour";th
    end do
    do while th < 0
       print er$(2)
```

```
   input "set hour";th
end do
input "set minute the realtime clock will
       start after you press <enter>";tm
do while tm > 59
    print er$(4)
input "set minute the realtime clock will
       start after you press <enter>";tm
end do
do while tm < 0
   print er$(2)
   input "set minute the realtime clock will
          start after you press <enter>";tm
end do
timer=0(start realtime clock)
input "boat lane's course";co
do while co > 360
   print er$(3):
   input "boat lane's course";co
end do
do while co < 0
   print er$(2)
   input "boat lane's course";co
end do
input "wave's speed";sp
do while sp < 2
   print er$(2)
```

```
        input "wave's speed";sp
      end do
 return


subroutine compute time difference
   tu = int(wx*.3/sp)
   tp = 60*(hh-th)+(mm-tm)
   if tp = 60/sp then
        zx = 36
        zy = 54
        a$ = "zer0 flag down"
        call print_message(zx,zy,a$)
   end if
   if int(tp-60/sp) = 5 then
        zx = 36
        zy = 54
        a$ = "zero flag up"
        call print_message(zx,zy,a$)
   end if
   tc = tu-tp
   if tc = 0 then
        a$ = "you are ontime"
        zy = 36
        zx = 152
        t$ = a$
        call print_message(zx,zy,a$)
```

```
  else if tc < 0 then
          t$ = "min.early"
       else
          t$="min.late"
       end if
       tc = int(abs(tc))
       if t$ = tt$ then
          zy = 36
          zx = 152
          a$ = str$(tc)
          call print_message(zx,zy,a$)
       end if
       if T$ <> tt$ then
          zy = 36
          zx = 152
          a$ = str$(tc) + t$
          call print_message(zx,zy,a$)
        end if
    end if
    tt$ = t$
return


subroutine error handler
  zx = 36
  zy = 54
```

```
  a$ = er$(em)
  call print_message(zx,zy,a$)
return


subroutine error table
   er$(1) = "out of screen   "
   er$(2) = "too small value "
   er$(3) = "greater than 360"
   er$(4) = "toc large value "
   er$(5) = "invalid input   "
return


subroutine read time
   ts = fix(timer/60)
   if tc <> ts then
      tc = ts
      if ts >= 60 then
         timer = 0
         tm = tm+1
         ts=ts-60
      end if
      if tm >= 60 then
         th = th+1
         tm = 0
```

```
      end if
      if th >= 24 then
         th = 0
      end if
    end if
return


subroutine print time
   a$ = str$(th) + str$(tm) + str$(ts) + " "
   zy = 36
   zx = 90
   call print_message(zx,zy,a$)
return


subroutine wave position
   zx = 78
   zy = 18
   a$ = " ?    "
   call print_message(zx,zy,a$)
   zx = 84
   call numeric_input(zx,zy)
   br = xv
   do while br > 360 or br < 0
        em = 6
```

```
      call error_handler(em)
      zx = 78
      zy = 18
      a$ = " ?    "
      call print_message(zx,zy,a$)
      zx = 84
      call numeric_input(zx,zy)
      br = xv
end do
zx = 78
zy = 27
a$ = " ?     "
call print_message(zx,zy,a$)
zx = 84
call numeric_input(zx,zy)
do while em = 4 or em = 6
      em = 0
      zx = 78
      zy = 27
      a$ = " ?     "
      call print_message(zx,zy,a$)
      zx = 84
      call numeric_input(zx,zy)
 end do
 rg = xv
 do while rg < 0
      em = 6
```

```
    call error_handler(em)
    zx = 78
    zy = 27
    a$ = " ?     "
    call print_message(zx,zy,a$)
    zx = 84
    call numeric_input(zx,zy)
end do
wh = (br-co)/180*3.141592654
wx = int(px-rg/10*cos(wh))
wy = int(py-rg/10*sin(wh))
if wx < o then
   wx=0
end if
if wx > 255 or wy > 191 or wy < 0 then
   em=1
   call error_handler(em)
return
end if
if fg = 0 then
    preset(wx,wy)
    fg=1
 else if fg = 1 then
    line(wx,wy)-(vx-vy),preset
    put(vx,vy)-(vx+24,vy+27),b,pset
 end if
 vx = wx
```

```
      vy = wy
      get(wx,wy)-(wx+z4,wy+z7),b,g
      zx = wx
      zy = wy
      a$ = "<"
      call print_message(zx,zy,a$)
   return


subroutine set time's scale
    sc = sp*.55555*2'20 sec
    k = 0
    do for i = 0 to 220 step sc
       if k/3-int(k/3) = 0 then
            a$ = str$(int(k/3))
            zx = i
            zy = 180
            call print_message(zx,zy,a$)
            line(i,132)-(i,176),preset end if
       line(i,172)-(i,176),preset
       k = k+1
    next i
return


subroutine plot grid
```

```
g = 132
do for i= 1 to 6
   line(0,g)-(255,g),preset
   g = g+5
   if g = 147 then
      g=162
   end if
next i
do for i = 0 to 255 step 20
   line(i,128)-(i,172),preset
   if i <> 0 then
      zx = i-12
      zy = 120
      a$ = str$(int(i/10))
      call print_message(zx,zy,a$)
   end if
next i
zx = px
zy = py
a$ = "*"
call print_message(zx,zy,a$)
line(0,128)-(20,80),preset
line(20,80)-(20,30),preset
sh$ = "shore"
zy = 36
do for i = 1 to 5
   zx = 0
```

```
      a$ = mid$(sh$,i,1)
      call print_message(zx,zy,a$)
      zy = zy+9
   next i
   draw "bm112,108;c0;e10;d4;r24;d12;l24;d4;h10"
   zx = 118
   zy = 105
   a$ = str$(co)
   call print_message(zx,zy,a$)
   zx = 60
   zy = 88
   a$ = "ship tc shore simulation"
   call print_message(zx,zy,a$)
return


subroutine numeric input
   call print_message(zx,zy,a$) 'input
   c$ = ""
   i=1
   A$ = inkey$
   if a$ <> chr$(13) then
      cl = len(c$)-1
      do while a$ = chr$(8)
         c$ = left$(c$,cl)
         i = i-1
```

```
        zx = zx-6
        A$ = inkey$
    end do
    do while a$ = ""
       call read_time
       A$ = inkey$
    end do
    if asc(a$) < 48 or asc(a$) > 57 then
       em=6
       call error_handler(em)
    else
       return
    end if
    c$ = c$+a$
    call print_message(zx,zy,a$)
    i = i+1
    if i > 5 then
       em = 4
       call error_handler(em)
       return
    end if
  end if
  xv = val(c$)
return
```

```
subroutine set character
   dim a(2), b(2), er$(6)
   (clear screen)
   print "please wait for loading characters !"
   z4 = 4
   z7 = 7
   z8 = 6
   z9 = 9
   do for z1 = 0 to 479
      read zc
      poke 31647+z1,zc
   next z1
return


subroutine print message
   l = len(a$)
   do for j = 1 to l
      zz$ = mid$(a$,j,1)
      call print_character(zx,zy,z4,z7,z8,zz$)
   next j
return


subroutine print character
   zc = asc(zz$)
   zc = zc-32
   z1 = zc*5+31647
```

```
    zq = varptr(a(0))
    poke zq, peek(z1)
    poke zq+1, peek(z1+1)
    poke zq+2, peek(z1+2)
    poke zq+3, peek(z1+3)
    poke zq+4, peek(z1+4)
    put(zx,zy)-(zx+z4,zy+z7),a,pset
    zx = zx+z8
return
```

APPENDIX B

PROGRAM LISTING.

```
 5 rem this program is  to simulate the amphibious operation
in ship-to- shore·movement phase
 10 'by lt. Chanck Hongnoi
 15 'october 4, 1982
 20 rem varible declaration
 25 'a$  = string buffer for display
 30 'c$  = numeric input buffer
 35 'gc$ = command used for control
 40 'gr$ = grid report buffer
 45 'q$  = quit command
 50 'sH$ = output the word "shore"
 55 't$  = time report storage
 60 'tt$ = previous time report output
 65 'zz$ = character of a$
 70 'bp  = offset bearing
 75 'br  = bearing wave input
 80 'cl  = lenght of c$
 85 'co  = lane's course
 90 'em  = error message pointer
 95 'fg  = flag of position plot
 100 'g  = varible grid plot
 105 'hh = h-hour
```

```
110 'i  = loop counter
115 'j  = loop counter
120 'k  = loop counter
125 'l  = length of a$
130 'mm = m-minute
135 'ph = direction in degree
140 'px = horizcntol axis of pcs.
145 'py = vertical axis of pcs.
150 'qq = quit flag
155 'rg = range from wave guide
160 'rn = offset range
165 'sc = time scale
170 'sp = speed of wave guide
175 'tc = difference of present time and predicted time
180 'tp = predicted time
185 'th = time in hour
190 'tm = time in minute
195 'ts = time in second
200 'tu = present time of wave
205 'vx = previous horizontal axis of wave
210 'vy = previous vertical axis of wave
215 'wh = relative bearing of wave in degree
220 'wx = horizcntal axis of wave location
225 'wy = vertical axis of wave location
230 'xv = numeric input from key board
235 'z4 = character width
240 'z7 = character high
```

```
245 'z8 = character gap
250 'z9 = line gap
255 'zc = character data
260 'zl = data pointer to read character
265 'zq = varible pointer of array "a"
270 'zx = horizontal location of alphanumeric output
275 'zy = vertical location of alphanumeric output

299 '*************************************************
300 rem master control program                       *
301 '*************************************************
305 clear 200,31646
310 gosub 1900'set character
315 gosub 1100  'error table
320 gosub 700   'input data
325 gosub 400   'offset
330 gosub 500   'echo input
335 gosub 1500  'set time scale
340 gosub 1600  'plot grid
345 gosub 1300  'wave position
350 gosub 1245  'print time
355 gosub 900   'compute time difference
360 gosub 600   'grid report
365 if qq<>1 then 345
370 q$ = inkey$:
    if q$ = "" then 370
375 if q$ <> "q" then 320
```

```
380 stop
385 end

400 '*****************************************
405 rem subroutine offset                      *
406 '*****************************************
410  input "the offset of  primary control ship from
reference point in bearing" ;bp
415 if bp > 360 then
       print er$(3):
       goto 410
420 if bp < 0 then
       print er$(2):
       goto 410
425 input "range from reference point";rn
430 ph = int(bp-co)/180*3.141592654
435 px = int(200-rn/10*cos(ph))
440 py = int(172-rn/10*sin(ph))
445 if px > 255 or px < 0 or py > 191 or py < 72 then
       print er$(1):
       goto 425
450 return

500 '*****************************************
505 rem subroutine echo input                  *
506 '*****************************************
510 pmode 4,1:
    screen 1,0:
```

```
    pcls 1
515 zx=0:
    zy=0:
    a$="wave's speed is" + str$(sp) + " wave's course is" +
str$(co):
    gosub 2000 'print message
520 zx = 0:
    zy = 9:
    a$ = "h-hour is" + str$(hh) + ":" + str$(mm):
    gosub 2000 'print message
525 zx=36:
    zy=18:
    a$="bearing":
    gosub 2000 :
    zx=36:
    zy=27:
    a$="range":
    gosub 2000 'print message
530 zx=36:
    zy=36:
    a$="time":
    gosub 2000 'print message
535 zx = 36:
    zy = 45:
    a$ = "grid position is":
    gosub 2000 'print message
540 zx = 114:
```

```
    zy = 18:
    a$ = "degrees":
    gosub 2000 'print message
545 zx = 114:
    zy = 27:
    a$ = "yards":
    gosub 2000 'print message
550 return

600 '*****************************************
605 rem subroutine grid report                 *
606 '*****************************************
610 if wy < 142 and wy >= 137 then
       gr$ = "r ":
       gc$ = "vector left 5   "
615 if wy < 162 and wy >= 142 then
       gr$ = "c ":
       gc$ = "                "
620 if wy < 167 and wy >= 162 then
       gr$ = "l ":
       gc$ = "vector right 5  "
625 if wy >= 167 then
       gr$ = "ll":
       gc$ = "vector right 10 "
630 if wy < 137 then
       gr$ = "rr":
       gc$ = "vector left 10  "
```

```
635 if wx <=0 then
       gr$ = "tcuch down":
       zx = 32:
       zy = 72:
       a$ = "press'q' to quit else continue !":
       gosub 2000 :
       qq = 1
640 zy = 45:
    zx = 138:
    a$ = gr$:
    gosub 2000 'print message
645 if gq$ <> gc$ then
       zy = 54:
       zx = 36:
       a$ = gc$:
       gosub 2000 :
       gq$ = gc$
650 return

700 '*****************************************
705 rem subroutine input data                  *
706 '*****************************************
710 cls
715 input"h-hour";hh
720 if hh > 24 then
       print er$(4):
       goto 715
```

```
725 if hh < 0 then
       print er$(2):
       goto 715
730 input "m-minute";mm
735 if mm > 60 then
       print er$(4):
       goto 730
740 if mm < 0 then
       print er$(2):
       goto 730
745 input "set hour";th
750 if th > 24 then
       print er$(4):
       goto 745
755 if th < 0 then
       print er$(2):
       goto 745
760 input "set  minute the realtime clock  will start after
you press <enter>";tm
765 if tm > 60 then
        print er$(4):
        goto 760
770 if tm<0 then
       print er$(2):
      goto 760
775 timer=0
780 input "boat lane's course";co
```

```
785 if co > 360 then
       print er$(3):
       goto 780
790 if co < 0 then
       print er$(2):
       goto 780
795 input "wave's speed";sp
800 if sp < 2 then
       print er$(2):
       goto 795
805 return

900 '*************************************************
905 rem subroutine compute time difference       *
906 '*************************************************
910 tu = int(wx*.3/sp)
915 tp = 60*(hh-th)+(mm-tm)
920 if tp = 60/sp then
       zx = 36:
       zy = 54:
       a$ = "zero flag down":
       gosub 2000 'print message
925 if int(tp-60/sp) = 5 then
       zx = 36:
       zy = 54:
       a$ = "zero flag up":
       gosub 2000 'print message
```

```
930 tc = tu-tp
935 if tc = 0 then
       a$ = "u are ontime":
       zy = 36:
       zx = 152:
       t$ = a$:
       gosub 2000:
       goto 960
940 if tc < 0 then
       t$ = " min.early "
    else t$=" min.late  "
945 tc = int(abs(tc))
950 if t$ = tt$ then
       zy = 36:
       zx = 152:
       a$ = str$(tc):
       gosub 2000 'print message
955 if T$ <> tt$ then
       zy = 36:
       zx = 152:
       a$ = str$(tc) + t$:
       gosub 2000 'print message
960 tt$ = t$
965 return

1000 '*****************************************
1005 rem subordinate error handler              *
```

```
1006 '*****************************************
1010 zx = 36:
     zy = 54
1015 a$ = er$(em)
1020 gosub 2000 'print message
1025 return

1000 '*****************************************
1105 rem subroutine error table                 *
1106 '*****************************************
1107 er$(1) = "cut of screen    "
1110 er$(2) = "too small value  "
1115 er$(3) = "greater than 360 "
1120 er$(4) = "too large value  "
1130 er$(5) = "invalid input    "
1135 return

1200 '*****************************************
1205 rem subroutine read time                   *
1206 '*****************************************
1210 ts = fix(timer/60)
1215 if tc = ts then 1240
1220 tc = ts
1225 if ts >= 60 then
        timer = 0:
        tm = tm+1:
        ts=ts-60
1230 if tm >= 60 then
```

```
        th = th+1:
        tm = 0
1235 if th >= 24 then
        th = 0
1240 return

1244 '*****************************************
1245 rem subroutine print time                  *
1246 '*****************************************
1250 a$ = str$(th) + str$(tm) + str$(ts) + " "
1255 zy = 36:
     zx = 90:
     gosub 2000 'print message
1260 return

1300 '*****************************************
1305 rem subroutine wave position               *
1306 '*****************************************
1310 zx = 78:
     zy = 18:
     a$ = " ?    ":
     gosub 2000:
     zx = 84:
     gosub 1815
1320 br = xv
1325 if br > 360 or br < 0 then
        em = 6:
        gosub 1000:
```

```
        goto 1310
1330    zx = 78:
        zy = 27:
        a$ = " ?     ":
        gosub 2000:
        zx = 84:
        gosub 1815
1335 if em = 4 or em = 6 then
        em = 0:
        goto 1330
1340    rg = xv
1345 if rg < 0 then
        em = 6:
        gosub 1000:
        goto 1330
1350 wh = (br-co)/180*3.14159265
1355 wx = int(px-rg/10*cos(wh))
1360 wy = int(py-rg/10*sin(wh))
1365 if wx < o then
        wx=0
1370 if wx > 255 or wy > 191 or wy < 0 then
        em=1:
        gosub 1000:
        return
1375 if fg = 0 then
        preset(wx,wy):
        fg=1:
```

```
        goto 1390
1380 if fg = 1 then
        line(wx,wy)-(vx-vy),preset
1385 put(vx,vy)-(vx+z4,vy+z7),b,pset
1390 vx = wx:
     vy = wy
1395 get(wx,wy)-(wx+z4,wy+z7),b,g
1400 zx = wx:
     zy = wy:
     a$ = "<":
     gosub 2000 'print message
1405 return

1500 '*****************************************
1505 rem subroutine set time's scale           *
1506 '*****************************************
1510 sc = sp*.55555*2
1515 k = 0
1520 for i = 0 to 220 step sc
1525    if k/3-int(k/3) = 0 then
           a$ = str$(int(k/3)):
           zx = i:
           zy = 180:
           gosub 2000:
           line(i,132)-(i,176),preset
1530    line(i,172)-(i,176),preset
1535    k = k+1
```

```
1540 next i
1545 return '

1600 '*****************************************
1605 rem subroutine plot grid                    *
1606 '*****************************************
1610 g = 132
1615 for i=1 to 6
1620    line(0,g)-(255,g),preset
1625    g = g+5
1630    if g = 147 then
           g=162
1635 next i
1640 for i = 0 to 255 step 20
1645    line(i,128)-(i,172),preset
1650    if i=0 then 1665
1655       zx = i-12:
           zy = 120:
           a$ = str$(int(i/10))
1660    gosub 2000 'print message
1665 next i
1670 zx = px:
     zy = py:
     a$ = "#":
     gosub 2000 'print message
1675 line(0,128)-(20,80),preset
1680 line(20,80)-(20,30),preset
```

```
1685 sh$ = "shore"
1690 zy = 36
1695 for i = 1 to 5
1700    zx = 0
1705    a$ = mid$(sh$,i,1)
1710    gosub 2000 'print message
1715    zy = zy+9
1720 next i
1725 draw "bm112,108;c0;e10;d4;r24;d12;l24;d4;h10"
1730 zx = 118:
     zy = 105:
     a$ = str$(co):
     gosub 2000 'print message
1735 zx = 60:
     zy = 88:
     a$ = "ship to shore simulation":
     gosub 2000 'print message
1740 return

1800 '*****************************************
1805 rem subroutine numeric input                *
1806 '*****************************************
1810 gosub 2000 'print message
1815 c$ = "":
     i=1
1820 A$ = inkey$:
     if a$ = chr$(13) then 1865
```

```
1825 cl = len(c$)-1
1830 if a$ = chr$(8) then
        c$ = left$(c$,cl):
        i = i-1:
        zx = zx-6:
        goto 1820
1835 if a$ = "" then
        gosub 1200:
        goto 1820
1840 if asc(a$) < 48 or asc(a$) > 57 then
        em=6:
        gosub 1000:
        goto 1870
1845 c$ = c$+a$
1850 gosub 2000 'print message
1855 i = i+1:
     if i > 5 then
        em = 4:
        gosub 1000:
        return
1860 goto 1820
1865 xv = val(c$)
1870 return

1900 '*************************************
1905 rem subroutine set character          *
1906 '*************************************
```

```
1910 dim a(2), b(2), er$(6)
1915 cls
1920 print "please wait for loading characters !"
1925 z4 = 4:
     z7 = 7:
     z8 = 6:
     z9 = 9
1930 for z1 = 0 to 479
1935 read zc
1940 poke 31647+z1,zc
1945 next z1
1950 return

2000 '*****************************************
2005 rem subroutine print message               *
2006 '*****************************************
2010 l = len(a$)
2015 for j = 1 to l
2020    zz$ = mid$(a$,j,1)
2025    gosub 2100 'print character
2030 next j
2035 return

2100 '*****************************************
2105 rem subroutine print character             *
2106 '*****************************************
2110 zc = asc(zz$)
2115 zc = zc-32
```

```
2125 zl = zc*5+31647
2130 zq = varptr(a(0))
2135 poke zq, peek(zl):
     poke zq+1, peek(zl+1):
     poke zq+2, peek(zl+2):
     poke zq+3, peek(zl+3):
     poke zq+4, peek(zl+4)
2140 put(zx,zy)-(zx+z4,zy+z7),a,pset
2145 zx = zx+z8
2146 '
2150 return

2155 '*****************************************
2200 rem character data area*
2205 '*****************************************
2232 data 255, 255, 255, 255, 255
2237 data 222, 247, 189, 255, 127
2242 data 173, 107, 95, 255, 255
2247 data 173, 65, 80, 86, 191
2252 data 220, 23, 29, 7, 127
2257 data 57, 187, 187, 179, 159
2262 data 186, 215, 117, 54, 95
2267 data 156, 238, 255, 255, 255
2272 data 238, 239, 123, 239, 191
2277 data 190, 251, 222, 238, 255
2282 data 218, 162, 8, 171, 127
2287 data 254, 246, 13, 239, 255
```

```
2292 data 255, 255, 57, 221, 255
2297 data 255, 254, 15, 255, 255
2302 data 255, 255, 255, 156, 255
2307 data 247, 187, 187, 189, 255
2312 data 139, 152, 163, 58, 63
2317 data 220, 247, 189, 238,63
2322 data 139, 189, 23, 188, 31
2327 data 139, 189, 159, 58, 63
2332 data 238, 106, 208, 119, 191
2337 data 3, 195, 239, 58, 63
2342 data 205, 222, 23, 58, 63
2347 data 7, 187, 187, 189, 255
2352 data 139, 157, 23, 58, 63
2357 data 139, 157, 15, 58, 63
2362 data 252, 231, 249, 207, 255
2367 data 156, 255, 57, 221, 255
2372 data 238, 238, 251, 239, 191
2377 data 255, 193, 240, 127, 255
2382 data 190, 251, 238, 238, 255
2387 data 139, 189, 221, 255, 127
2392 data 139, 189, 37, 42, 63
2397 data 221, 92, 224, 57, 223
2402 data 13, 173, 27, 88, 63
2407 data 139, 158, 247, 186, 63
2412 data 13, 173, 107, 88, 63
2417 data 3, 222, 55, 188, 31
2422 data 3, 222, 55, 189, 255
```

```
2427 data 131, 222, 199, 58, 63
2432 data 115, 156, 7, 57, 223,
2437 data 142, 247, 189, 238, 63
2442 data 247, 189, 239, 58, 63
2447 data 115, 86, 117, 181, 223
2452 data 123, 222, 247, 188, 31
2457 data 113, 20, 167, 57, 223
2462 data 113, 148, 199, 57, 223
2467 data 3, 156, 231, 56, 31
2472 data 11, 156, 23, 189, 255
2477 data 139, 156, 229, 54, 95
2482 data 11, 156, 21, 181, 223
2487 data 139, 159, 31, 58, 63
2492 data 6, 247, 189, 239, 127
2497 data 115, 156, 231, 58, 63
2502 data 115, 157, 90, 239, 127
2507 data 115, 156, 229, 17, 223
2512 data 115, 171, 186, 185, 223
2517 data 115, 171, 189, 239,127
2522 data 7, 187, 187, 188, 31
2527 data 141, 239, 123, 222, 63
2532 data 123, 239, 190, 251, 223
2537 data 143, 123, 222, 246, 63
2542 data 220, 85, 189, 239, 127
2547 data 254, 238, 11, 239, 255
2552 data 206, 119, 223, 255, 255
2557 data 255, 227, 232, 58, 31
```

```
2562 data 123, 210, 103, 25, 63
2567 data 255, 226, 231, 186, 63
2572 data 247, 164, 199, 50, 95
2577 data 255, 226, 224, 62, 63
2582 data 238, 183, 29, 239, 127
2587 data 252, 152, 201, 121, 209
2592 data 123, 222, 147, 57, 223
2597 data 254, 255, 61, 238, 63
2602 data 255, 191, 239, 121, 209
2607 data 123, 218, 179, 173, 191
2612 data 158, 247, 189, 238, 63
2617 data 255, 234, 165, 57, 223
2622 data 255, 210, 103, 57, 223
2627 data 255, 226, 231, 58, 63
2632 data 250, 76, 227, 37, 239
2637 data 252, 152, 230, 75, 222
2642 data 255, 210, 103, 189, 255
2647 data 255, 224, 248, 248, 63
2652 data 222, 193, 189, 235, 191
2657 data 255, 220, 231, 58, 63
2662 data 255, 220, 231, 87, 127
2667 data 255, 220, 229, 42, 191
2672 data 255, 221, 93, 213, 223
2577 data 251, 156, 232, 121, 209
2682 data 255, 193, 221, 220, 31
2687 data 238, 247, 125, 239, 191
2692 data 222, 247, 253, 239, 127
```

```
2697 data 190, 247, 221, 238, 255
2702 data 186, 187, 255, 255, 255
2707 data 0, 0, 0, 0, 0,
```

APPENDIX C

COMMAND SUMMARY.

THE COMMANDS USED IN COMPUTER.

WORD	PURPOSE
ABS	Computers absolute value.
ASC	Returns ASCII code of first character of specified string.
ATN	Returns arc tangent in radians
AUDIO	Connects or disconnects cassette output to TV speaker.
CHR$	Returns character for ASCII, control, or graphics code.
CIRCLE	Draws a circle with center at point(X,Y) with a radius of r, of a specified colour c, with height/width ratio(hw) of 0-4 Circle can begin and end at specified point (0-1).
CLEAR	Reverses n bytes of string storage space. Initializes variables. Specifies highest BASIC address.
CLOAD	Load specified program file from cassette. If filename is not

	specified, first file encountered is loaded. Filename must be eight characters/spaces or less.
CLOADM	Loads machine language program from cassette. An offset address to add the loading address may be specified.
CLOSE	Closes open file or devices.
CLS	Clears display to specified color. If color is not specified, green is is used.
COLOR	Sets forground and background color.
CONT	Continues program execution after pressing BREAK or using STOP statement.
COS	Returns cosine of angle measured inradians.
CSAVE	Saves program on cassette(program name must be eight character/ spaces or less). For ASCII format, use "A."
CSAVEM	Writes out a machine-language file.
DATA	Stores data in your program. Use READ to assign this data to variables.
DEF FN	Defines numeric function.
DEFUSR	defines entry point for USR function.
DEL	Allows deletion of program lines. DEL- Delete entire program. DEL n Deletes specified line n.

DEL n- Deletes all lines past n.

DEL -n Deletes all lines up to n.

DELn-n Deletes all lines between n and n.

DIM Dimensions one or more arrays.

A 4K system allows only one dimension.

A 16K allows multi-dimensional arrays.

DLOADM Loads machine-language program at specified baud.

0 = 300 baud 1 = 1200 baud

DRAW Draws a line beginning at specified starting point of specified length of specified color. Will also draw to scale, draw blank lines, draw non-updated lines and execute substrings. If starting point is not specified, (128,96) or last DRAW position is used.

EDIT Allows editing of program line.

nC Changes n number of characters.

nD Deletes n number of characters.

I Allows insertion of new character.

H Deletes rest of line and allows insert.

L Lists current line and continues edit.

	nSc Searches for nth occurrence of character c.
	X Extends line.
	SHIFT ¬ Escape from subcommand.
	n SPACEBAR Moves cursor n spaces to right.
	n < Moves cursor n spaces to left.
END	Ends program.
EOF	Returns FALSE(0) if there is more data; TRUE(-1) if no data is in the specified file. For cassette files, f = -1; for keyboard files, f=0.
EXEC	Transfers control to machine-language programs at specified address. If address is omitted, control is transferred to address set in last CLOADM.
EXP	Return natural exponential of number (e number)
Exponential	Raises number to specified power
FIX	Return truncated value. All integers are truncated.
FOR..TO STEP/ NEXT	Creates a loop in program which the computer must repeat from the first number to the last number you specify. Use step to specify how much to increment the number each time through the loop. If

	you omit STEP one is used.
GET	Read the graphic contents of a ractangle into an array for future use by PUT.
GOSUB	Sends the computer to subroutine beginning at specified line number.
GOTO	Sends the computer to the specified line number.
HEX$	Compute hexadecimal value
IF/THEN	Test the relationship. If it is true, the computer execute the instruction following THEN.
INKEY$	Strobe the keyboard and returns the key being pressed.
INPUT	Cause the computer to stop and await input from the keyboard
INPUT#-1	Input data from cassette
INSTR	Search for the first occurance
INSTR$(5,X$,Y$)	of string Y$ in string X$ and returns the position at which the match is found.
INT	Convert a number to an integer.
JOYSTK	Returns the horizontal or vertical coordinate of the left or right joystick: 0 =horizontal, left joystick 1 = vertical, right joystick

2 = horizontal,right joystick

3 = vertical, left joystick

LEFT$ Return the left potion of the string.

LEN Retuns the number of character in a string.

LET Assign value to variable

LIST Lists specified lines or entire program on the screen LIST 50-100

LINE Draw a line from the start point to end point if start point is omited, (128,96) or the last end point is used. PSET selects fcreground color and PRESET select background color. ,B draws a box with start and end points as the corners(instead of a line). ,BF will fill in the box.

LINE INPUT Input line from keyboard.

LOG Returns natural logarithm.

MEM Finds amount of free memory.

MID$ Returns a substring if another string. If length option is omitted, the entire string right cf position is returned.

MID$ Replaces a portion of one string with another string.

MOTOR Turns cassette ON or OFF.

NEW	Erases everything in memory.
ON...GOSUB	Multi-way branch to specified subroutines.
ON...GOTO	Multi-way branch to specified lines.
OPEN	Open file(f) at Screen or Keyboard(0), Cassette(-1), Printer(-2) for input(I) or (0).
PAINT	Paints graphic screen starting at point (X,Y) with specified color. Stops at border of specified color.
PCLEAR	Reserves n number of graphic memory pages.
PCLS	Clears screen with specified color. If color code is omitted, current background color is used.
PCOPY	Copies graphic from source to destination page.
PEEK	Returns the contents in the memory location you specified.
PLAY	Plays music of specified note (A-G or 1-12)octave(0), volume (V) note length(L), tempo(T), pause(P), and allows execution of substrings. Also sharps(# or +) or flats(-).
PMODE	Selects resolution and memory page to start on.
POINT	Tests whether specified graphics cell is on or off, X(horizontal)

=0-31. The value returned is -1 if the cell is in the character mode, 0 if it is off, or the color code if it is on.

POKE — Puts values into specified memory location.

POS — Returns current cursor position.

PPOINT — Tests whether specified graphics cell is on ot off and returns color code of specified cell.

PRESET — Sets to background color.

PRINT — Prints specified message on the screen.

PRINT#-1 — Writes data to cassette.

PRINT#-2 — Prints an item or list of items on the printer.

PRINT TAB — Moves the cursor to specified column position and prints.

PRINT USING — Prints numbers in specified format.

- # Formats numbers.
- . Decimal point.
- , Displays comma to left of every third character.
- ** Fills leading spaces with asterisks.
- $ Places $ ahead.

	$$ Floating dollar sign.
	**$ Floating dollar sign
	+ In first position, causes sign to be printed. In last position, causes sign to be printed after the number Exponential format.
	- Minus sign after negative numbers. Returns first string charater.
%spaces%	String field; length of field is nember of spaces plus 2.
PRINT@	Prints specified message at specific screen location.
PSET	Set specified point to specified if C is omited, foreground color is used.
PUT	Stores graphics from an array V,PSET onto the screen. (Array rectangle size must match GET rectangle size.)
READ	Read the next item in DATA lineREAD A$ and assign it to specified READ C,B variable.
REM or'	Allow insertion of comment in progream line. Everything after REM is ignored by Computer.
RENUM	Allows program line renumbering.
RESET	Erases dot SET at specified location.
RESTORE	Sets the Computer's pointer

	back to first item on the line.
RETURN	Returns the computer from subroutine to the BASIC word following
RIGHT$	Returns right potion of string.
RND	Returns a pseudo -random integer between one and specified number which must be greater than one.
RUN	Executes the program.
SCREEN	Selects ether graphics or text screen and color set
SET	Sets a dot at specified text screen location to specified color.
SGN	Returns sign of specified numeric expression: -1 for argument is negative 0 if argument is 0 +1if argument is positive
SKIPF	Skip file to end of next program on cassette tape, or to end of specified program.
SIN	Returns sine of angle measured in radians.
SOUND	Sounds specified tone for specified duration.
STOP	Stops execution of program.
STRING$	Returns a string of characters (of specified length) which are identified by ASCII code or the

	first character of the string.
STR$	Convert a numeric expression to a string.
SQR	Returns the square root of number.
TAN	Returns tangent of angle measured in radians.
TIMER	Returns contents or allows setting of timer (0-65535)
TROFF	Turns off program tracer.
TRON	Turns on program tracer.
USERn	Calls user's machine language subroutine.
VAL	Converts a string to number.
VARPTR	Returns "pointer" address for the specified variable.

LIST OF REFERENCES

1. Ladd, J. D., Assault from the Sea 1939-45 Hippocrene Books, Inc., 1976

2. Tomlinson, George E., "Doctrine for Landing Forces June 1963", Coordinator marine corps landing force development activities., 1963

3. Tandy company "Getting Start with Basic.", 1979

4. Tandy company "Going Ahead with Extended Basic.", 1979

5. Department of the Army, Doctrine for Amphibious Operations, 1962

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 54 Department of Administrative Science Naval Postgraduate School Monterey, California 93940	1
4.	LT Chanok Hongnoi RTN. Royal Thai Navy BKK. 10600, Thailand	1